MODULAR LANGUAGE TRANSLATION SYSTEM

Technical Field

This invention relates to automated natural language translation systems in which different user interface modules can inter-operate with different translation engine modules. For example, a single user interface module could reside on a computer and could invoke any one of a variety of available translation engine modules residing on one or more other computers located across a network.

Background Information

Machine Translation (hereinafter "MT") generally describes an area of computer applications that assist or automate the translation of text from one natural (spoken) language to another. The idea of decoding natural languages through mathematical techniques became a reality in the Cold-War period after World War II. The requirements of the intelligence community drove much of the early MT systems and research. Early MT systems typically used literal translation (word for word translation), without the use of linguistic rules.

The demand for MT systems remains high in part because of the increased use and importance of the Internet and other networked computer systems. Users worldwide regularly can exchange information with a speed and convenience not possible previously. This information, however, often must be translated to a natural language in which the user is fluent. Business users in particular often need to translate documents.

In general, current MT systems have user interfaces that differ depending on which source language is being translated to which target language. Thus, it is difficult for users to learn the interfaces or to translate rapidly or automatically a document into more than one target language.

For example, a certain MT system might support a translation option such as "spell check," which automatically scans a document before translation for spelling errors, whereas a different MT system might not support this option. Consequently, the user must tailor every request to translate a document to the specific options and requirements of the particular MT system being employed. Because each MT system can provide many different features and options, the user must learn and remember which MT systems are capable of particular options.

Moreover, because different MT systems often refer to the same concept by different terms, it becomes difficult for a user to set up a "standard" translation request. This lack of standardization makes it difficult for users to translate various documents.

Summary of the Invention

The present invention provides a translation system that allows a user of any one of a variety of different user interfaces to be able to send translation requests to and receive responses from any one of a variety of different translation engines. That is, a user of the present invention who is familiar with the user interface for a first type of MT system, such as one that translates from Japanese to English, can use that user interface to get translations from the translation engine of a second type of MT system, such as a Russian-to-English system, without having to learn the particularities of the second system and its interface. In addition, the present invention makes it easier for software developers to write applications that can communicate with a multitude of different MT systems. Further, the present invention allows a particular MT system to be able to process translation requests from any one of a number of other MT systems.

In one disclosed embodiment, the present invention uses a distributed object protocol, such as the Component Object Model (COM) or the Common Object Request Broker Architecture (CORBA), for all communications between a client and a translation engine. The client and the translation engine each have an interface defined by an interface definition language (IDL). All translation requests sent by the client and all responses generated by the translation engine are sent in accordance with the IDL that defines those interfaces. Therefore, the client interface and translation engine will be able to communicate regardless of what machine they are running on (as long as the machines are connected, such as by a network), what operating system the machine is running (as long as it supports the distributed object protocol employed), or the computer language in which the client or translation engine is written (as long as the language system supports the distributed object protocol).

By using a distributed object protocol to standardize the way in which user interfaces talk to translation engines, as proposed in the present invention, developers can design MT systems in a "modular" fashion. This benefits both the developers and users of MT systems. Users who purchase and familiarize themselves with an MT system for one type of language, such as English to Spanish, can use the interface from that system to communicate with an entirely different MT system, such as English to French, as needed without having to re-learn a new system.

In general, in one aspect, the invention features a translation system having a client and a translation engine. The client sends a translation request comprising text to be translated from a first language to a second language using a distributed object protocol. The client receives a response to the request using the distributed object protocol. The translation engine receives the translation request using the distributed object protocol and generates the response corresponding to a translation of the text from the first language to the second language, and sends the response to the client using the distributed object protocol.

Embodiments of this aspect of the invention can include the following features. The translation request and the response can be sent in accordance with an IDL, the IDL defining an interface at the client and an interface at the translation engine. The distributed object protocol used by the client and the translation engine to communicate can support a plurality of user interfaces. The distributed object protocol can operate in accordance with the Component Object Model (COM) standard or the Common Object Request Broker Architecture (CORBA) standard. The system can include a notification mechanism having an interface defined by an IDL by which the translation engine can inform the client of errors that occur during translation. The system can include a registration mechanism having an interface defined by an IDL by which the translation engine can register itself to be found by the client and can inform the client of components such as user dictionaries that are available with the translation engine. The client and the translation engine can be located on the same computer, or on different computers connected to each other. For example, the client and the translation engine can communicate over a computer network. The text to be translated can be represented by a character set that is capable of including characters used in a multitude of languages, such as UNICODE.

In general, in another aspect, the invention features a translation method in which a translation request is sent from a client to a translation engine using a distributed object protocol. The translation request comprises text to be translated from a first language to a second language. At the translation engine, the request is received using the distributed object protocol, and a translation is performed in response to the received request. The translation is included in a response to the request, and the response is sent from the translation engine to the client using the distributed object protocol.

Embodiments of this aspect of the invention can include the following features. For example, the step of sending a translation request from the client can comprise sending a translation request from the client to the translation engine in accordance with an IDL, where the IDL defines an interface at the client and at the translation engine. The translation method can include the step of providing a registration mechanism by which the translation engine can register itself to be found by the client, the registration mechanism having an interface defined by an IDL. The translation method can include the step of scanning the translation request for irregularities in the text, such as misspellings or undefined words, prior to sending it to the translation engine for translation. The translation method can include the step of providing to the client information regarding the progress of the translation by the translation engine.

In general, in still other aspects, the invention features a translation system comprising either a plurality of translation engines and a client or a translation engine and a plurality of clients. In either case, the single unit can inter-operate with any one of the plurality of units, and they communicate using a distributed object protocol such as COM or CORBA. In the first embodiment, there can be one or more clients, and in the second embodiment, there likewise can be one or more translation engines. In one embodiment, each of the one or more clients and each of the one or more translation engines has an interface define by an IDL, so that all communication between a client and a translation engine will occur in accordance with the IDL. In one embodiment, a translation engine can maintain separate contexts for each translation request that it receives, enabling the translation engine to translate each request in accordance with particular translation format that might be associated with the translation request.

The foregoing and other options, aspects, features, and advantages of the invention will become more apparent from the following detailed description of the invention when taken in conjunction with the accompanying drawings.

Brief Description of the Drawings

In the drawings, like reference characters generally refer to the same parts throughout the different views. Also, the drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention.

FIG. 1 is a block diagram of a modular language translation system in accordance with the invention.

FIG. 2 is a block diagram of a particular embodiment of the modular language translation system of FIG. 1 in accordance with the invention.

FIG. 3 is another block diagram of a modular language translation system of the invention.

FIG. 4 is a block diagram of a modular language translation system having multiple translation engines, in accordance with one embodiment of the invention.

FIG. 5 is a block diagram of a modular language translation system having multiple clients, in accordance with another embodiment of the invention.

FIG. 6 is a flow diagram illustrating the operation of the system of FIG. 2.

FIG. 7 is a flow diagram illustrating in more detail the step of sending a translation request shown in FIG. 6.

FIG. 8 is a flow diagram illustrating in more detail the translating step of translating shown in FIG. 6.

Description

Referring to FIGS. 1 and 2, in one embodiment a language translation system 10 according to the invention includes a client 11 and a translation engine 16. The client 11 can be any type of system (e.g., a user interface) capable of sending a translation request in accordance with a distributed object protocol to the translation engine 16 and receiving a response from the translation engine 16 in accordance with the distributed object protocol. (The distributed object protocol is explained more fully below.) For example, the client 11 could be a proxy server operating in accordance with a protocol such as Gopher, File Transfer Protocol (ftp), HyperText Transport Protocol (http), Simple Network Management Protocol (SNMP), and Simple Mail Transfer Protocol (SMTP), that calls a translation engine 16 to instantly translate content downloaded from a computer network such as the world-wide-web (WWW). In another embodiment, a client 11 could be a "web-crawler" or "web-robot" that searches the WWW for content and generically translates the content into a common language. The client 11 can be composed entirely of hardwired logic circuitry, but it preferably is realized mainly in software executing on a general purpose computer such as a PC or workstation.

In one embodiment, the client uses a character set that includes characters used in a plurality of languages, such as the UNICODE character set, which is an international character encoding standard that permits computers in one world language community to "talk" with those in another language community. For example, a client in one embodiment is a word processing application running on a computer. Alternatively, other software applications, such as spreadsheets, database programs, web browsers, etc., can be a client 11. The client 11 has a client interface 94 that is specified to be in accordance with the distributed object protocol. Preferably, this client interface 94 is defined by an interface definition language (IDL), but one skilled in the art will recognize that other mechanisms can be used to define the client interface 94. Software developers commonly work with this interface. In addition, in another embodiment, a client has a user interface 24 enabling the user of the application to interact with that application. The user interface 24 of the application or program can have a translation service that accepts translation requests. For example, in a script-enabled application such as MICROSOFT WORD 7.0, a WORD document could contain Visual Basic script that can transmit a translation request using a distributed object protocol.

In one embodiment, the translation request includes text to be translated from one natural language to another. The translation request can also include or alternatively can include information directing the translation engine how it should perform the translation, information relating to the particular text to be translated, information relating to the order of translation, or information relating to a particular dictionary to use during translation. For example, the information included with the translation request may include part-of-speech settings (indicating the particular parts of speech of one or more words in the text); translation hints; indicators of sentence boundaries; translation guidelines; indications for types of words to translate in a particular fashion, or indicators of discontinuities in the text. The information further could include information relating to the desired translation preferences indicating preferred dictionaries, alternate word settings, alternate translations, and alternate sentence translations, as well as indications as to what parts of the text to which the options should be applied. These examples of information are not intended to be limiting; one skilled in the art will recognize that many different types of translation preferences and options may be desirable to be included in the translation request. The information also can include markups such as Hypertext Markup Language ("HTML") and Standard Generalized Markup Language ("SGML") markup; Rich Text Format ("RTF") markup; and Nontypesetting Runoff ("NROFF") markup.

In some embodiments, systems other than general purpose computers are employed to realize the client 11, including palmtop computers, personal organizers, mainframe computers, and fax machines. Any system capable of communicating in accordance with the distributed object protocol is intended to be within the scope of the invention. In one embodiment, it is preferred that the distributed object protocol of the present invention support a plurality of user interfaces, computers, applications, etc. It is also preferred that an IDL defines an interface at each system used to realize the client 11.

Referring to the embodiments illustrated FIGS. 1 - 3, the translation engine 16 comprises a system that can receive the translation request in accordance with the distributed object protocol and it, like the client 11, preferably is realized mainly in software executing on a general purpose computer. Also, like the client 11, the translation engine 16 in this embodiment has a translation engine interface 96 that is specified to be in accordance with the distributed object protocol. Preferably, the translation engine interface 96 is defined by an IDL, although other mechanisms can be used to define the translation engine interface 96, as will be appreciated by those skilled in the art. It should be understood, however, that the translation engine 16 and client 11 could each be realized in any machine, apparatus, or article of manufacture capable of communicating using a distributed object protocol and capable of having an interface specified to be in accordance with a distributed object protocol. The translation engine 16 performs translation operations on the source text, in conjunction with data in storage 18. Commonly, data in storage 18 corresponds to all sorts of information used or useful in performing the translation including one or more dictionaries, domain keywords, grammar rules, other databases, etc. The translation engine 16 can be located on the same computer as the client 11 or on a separate computer in communication with the client 11.

After the translation engine 16 receives the translation request from the client 11, the translation engine 16 translation engine 16 translates the text from the first language to the second language. The translation engine 16 then uses this translated text to generate a response to send back to the client 11. The translation engine 16 sends the response to the client 11 using a distributed object protocol.

Referring to FIG. 2, which illustrates a particular embodiment of the invention, the translation system 10 also preferably includes a registry 84 by which client 11 can locate available translation engines 16. For example, when client 11 wants to submit a translation request, the client 11 can check the registry 84 to find out what translation engines are accessible to the client 11. In one embodiment this checking includes searching for translation engines 16 installed at the client 11. In another embodiment, the registry 84 searches across a computer network such as the

WWW to locate other translation engines 16 that operate in accordance with a distributed object protocol. In still another embodiment the registry 84 also provides information about options and/or components available on each located translation engine 16. Typically, these components include one or more of the following: user dictionaries, syntax dictionaries, user dictionary editors, dictionary browsers, secondary translation engines, alternate word finders, alternate sentence translators, sentence boundary finders, asynchronous (background) translators, synchronous translators, fuzzy translation matching, part-of-speech settings, selective translation, spell-checkers, and translation memory archives.

A translation engine 16 can have associated with it one or more translation options 86. A translation option 86 corresponds to a "global" option that applies to any translation request sent to the translation engine 16. For example, a translation option 86 might be an operation that scans for undefined words. Thus, for any translation request that a client 11 sends to a translation engine 16 having this option, the translation engine will scan text of the translation request for undefined words. In addition, the registry 84 listing the translation engine 16 having this option could indicate as part of the registry 84 that this translation option is available at this translation engine 16.

The translation request that the client 11 sends to the translation engine 16 typically comprises at least some text to be translated from a first natural language to a second natural language. For example, the translation request could comprise some English text that needs to be translated into Japanese. It should be understood that the invention is not limited to any particular pairs of languages. As noted above, the translation request can comprise information other than text to be translated, such as information directing the translation engine 16 to perform the translation in a particular manner or in a particular format. For example, the translation request can comprise a translation preference directing the translation engine 16 to translate in accordance with a particular user dictionary 38. Use of particular user dictionary 38 may be desirable in examples where the text to be translated relates to a particular domain of interest (such as technical or financial); use of a technical or financial dictionary helps to ensure accuracy of translating words that have specific meanings in those domains.

The information discussed above directing the translation engine 16 to perform the translation in a particular way, generally referred to as translation preferences, can be handled by the translation engine 16 in several ways. In one embodiment, the translation engine uses one or more translation preferences objects 88 that are responsive to information in the translation request relating to a desired format of the translation. In another embodiment, the translation system 10 can further include a preferences editor 26 that permits the client to set the preferences that will be provided to the translation engine 16 as a translation preferences object.

The translation request can be generated in a number of well-known ways. FIG. 3 illustrates one embodiment of the invention used with a system that can provide text to be translated and includes the client 11 having an input interface 12, a display 20, an output interface 14, and a user input device 22. In some embodiments, the user input interface 12 can be a keyboard, a mouse, touchscreen, or light pen. It should be understood that any user input device that a client 11 can use can be used as a user input interface 12. For example, in one embodiment the input interface 12 comprises a connection to a source that can provide text to be translated, such as a modem, serial line, web page, or another system in communication with the client 11. In some embodiments, the display 20 comprises a computer display or printer. Any apparatus or system capable of communicating information to the client 11 may be used as a display. The output interface 14 communicates the final translation of the source text in the second language, where the translation comes from the response sent by the translation engine 16. The output interface 14 may comprise a printer, fax machine, a voice interface, an electronic interface, such as a modem or serial line, or it may include other means for communicating the text to the end user.

The client 11 and translation engine 16 can be interconnected in any number of well-known ways to permit the client 11 to send a translation request to the translation engine 16. For example, in one embodiment, the client 11 and translation engine 16 are located on the same computer system. In another embodiment, the client 11 and translation engine 16 are located on different computers capable of communicating with each other. In another embodiment, this communication occurs via a computer network, such as the Internet. In still other embodiments, communication occurs via a wireless communication system, through coaxial cable such as a cable TV line, through a high speed communications link, or through a telephone line.

In addition, in still another embodiment, the client 11 and translation engine 16 could share a thin (narrowly defined) interface, wherein the thin interface is flexible enough to allow the client 11 and translation engine 16 to communicate, but supports only a limited number of Application Program Interfaces (API).

The distributed object protocol with which the client 11 and translation engine 16 communicate is a standard governing the creation of and communication between objects and distributed objects (objects that communicate with objects on other network nodes). Distributed object computing simplifies system development and maintenance by creating reusable pieces of code (commonly known as objects) and by allowing objects installed on different computers to communicate over a network. A software object is a collection of related function (or intelligence) and the function's (or intelligence's) associated state. A set of distributed objects, such as a set including all of the objects illustrated in Fig. 2 (the client 11, translation engine 16, dictionary browser 46, etc.), is a collection of independently operating nodes. In one embodiment, each node comprises an independent process. In another embodiment, each node has independent storage. However, it should be understood that a node need not comprise an independent process to be usable with the invention. In some embodiments, each node is on a separate and independent computer, on the same computer as all other nodes, or distributed among two or more computers.

A number of different distributed object protocols are presently available and could be used with the present invention. These include, for example, the Microsoft Component Object Model (COM), the Microsoft Distributed Component Object Model (DCOM), VisualWorks' Distributed Smalltalk, JAVA RML, and the Object Management Groups's Common Object Request Broker Architecture (CORBA) with its Internet Inter-Orb Protocol (IIOP). In the disclosed embodiment, COM and CORBA are the preferred distributed object protocols. COM presently is the defacto standard for building component architectures.

Under a distributed object protocol, objects on one node know the names of objects on other nodes. Handles to the distributed objects (also called interface pointers or object references) can be passed from machine to machine, program to program, and process to process. Any entity that can access a handle to a distributed object can call methods on it (calling methods is synonymous with sending a message). Thus, objects on one node can directly send messages to objects on other nodes, and objects on one node can freely migrate to another node. One way of ensuring that objects operating in accordance with a distributed object protocol can communicate with each other is by specifying on interface on each object using an interface definition language (IDL). When interfaces are specified using an IDL, the objects can communicate with each other even if the objects themselves are written in different programming languages, by different vendors, or as different versions of the same program.

Thus, an advantage of the disclosed system, which uses a distributed object protocol and has interfaces defined by IDLs, is interoperability. Clients and translation engines developed in accordance with the present invention can be treated as interoperable software modules, instead of as incompatible proprietary systems, such as are presently being employed in conventional language translation systems. Further, clients and translation engines developed in accordance with the present invention will be language, vendor, and version neutral. Thus, using this invention, a software developer can to design a translation API that works with multiple providers of a translation service transparently; that is, without any special knowledge of which provider or implementation is in use. Using the present invention, the provider of a translation service can express new, enhanced, or unique translation components to potential consumers in a standard manner, without worries that new versions will be incompatible with consumer's systems.

The problems faced by a software developer who wants to be able to develop a user interface (client) capable of using multiple translation engines are somewhat akin to the user of a personal computer who wants to be able to send a document to any one of a number of printers from any one of a number of applications. Such a user does not want to have to learn the various names and options that the applications and/or the printer manufacturers have given to printing options. To simplify this task, developers of desktop applications such as word processors and spreadsheets established standard "printer drivers" that make the task of conforming to a particular printer's needs transparent to the user. The user need only be familiar with the particular print interface of the application he or she is using. The user merely pulls up the standard "print" menu from the application's interface, selects from a set of standard print options (such as "double-sided"), and selects the desired printer. Users can select default options and/or default printers with confidence, knowing that the application and the printer will communicate to accomplish the desired functions or will alert the user when functions are not available. Users of a multitude of applications, therefore, are interoperable with a multitude of printers because the printers and applications act as objects defined by a standard protocol. By requiring that translation engines and clients be defined by a standard protocol, such as a distributed object protocol, the present invention enables clients and translation engines to be interoperable.

Referring again to FIG. 2, some of the options and features that one embodiment of a translation system 10 in accordance with the invention might include are shown. The translation system 10 comprises: a client 11 having a user interface 24, a translation engine 16 comprising a translation preferences server 28, synchronous translator 30, asynchronous translator 32, alternate word server 34, and translation server 36, a user dictionary editor 44, a dictionary browser 46, a preferences editor 26, a user dictionary 38, a syntax dictionary 40, and a base dictionary 42, part-of-speech registry 90, registry 84, alternate sentence server 92, translation option 86, and translation preference object 88.

The user interface 24 has a translation service for accepting translation requests. In one embodiment, the user interface 24 is a translation service comprising a standard set of fields into which the client 11 provides text to be translated and/or translation options. The user interface 24 need not be located on the same computer as the client. A user interface 24 might exist at a remote location accessible to the client 11, such as on a site accessible by a web browser.

In the disclosed embodiment of FIG. 2, the user interface 24 will be the same regardless of the translation request that a client 11 wants to make. That is, regardless of the application or system, a consistent user interface 24 is presented to the client. The user interface 24 permits a client to specify options that affect the translation performed by the translation engine 16. In addition, a client 11 using the user interface 24 can use the user interface 24 to tell the user of the system 10 if the translation engine 16 supports a particular translation preference, such as spell-checking. The client 11 sends a message to a preferences editor 26, which communicates with the translation preferences server 28 to see if the translation preferences that have been selected are available at the translation engine 16. Preferably, the preferences editor 26 has an interface defined by an IDL. The preferences editor 26 then notifies the client 11 about the availability of the translation preference so that the client 11 can formulate the translation request in accordance with this information and also provide this information to the user interface 24. When the translation engine 16 receives the translation request sent by the client 11 in accordance with the translation preferences, the translation server 36 receives the preference information and ensures that the translation engine 16 translates in accordance with it.

As an example of the use of preferences, consider an example of requesting a particular user dictionary. When the client 11 interacts with the preferences editor 26, the preferences editor 26 will display to the client 11 only the user dictionaries 38 available to the translation engine 16. If the client 11 selects a particular user dictionary 38, then the preferences editor 26 ensures that each time the client 11 sends a translation request to the translation engine, that the translation preferences object 88 reflects the preferred dictionary that the client 11 has selected. In one embodiment, the preferred dictionary is used until the client 11 resets the preference. Thus, the translation preferences may operate similarly to preferences used in other applications such as web browsers.

As an example of the use of options, consider the example of a spell-checker. If spell-checking is not available for translator 16, then when the user interface 24 is displayed to the user of the system, the spell-checking option will be displayed in a way that indicates that the user cannot select it as an option. For example, a selectable box for spell-checking on the user interface 24 might be "grayed out," indicating to the user that the user cannot select that option. However, if spell-checking is available at translation engine 16, then the user interface 24 will permit the user to select it as an option. Such a technique permits the client 11 and user interface 24 to be interoperable with translation engines 16 having different types of options. This type of feature can be used with components displayed in the registry 84 as well.

The embodiment of FIG. 2 provides other features helpful to a user seeking to translate text. The synchronous translator 30 is used if translation needs are simple. Preferably, the synchronous translator is implemented such that it has an interface defined by an IDL. It provides immediate translation. When the client 11 sends a translation request that requests use of the synchronous translator 30, the client waits for the translation engine 16 to send a response containing the translation of the text.

For example, an end user application, such as a word processing application, could contain script that calls for instant translation of a portion of text. Such script (typically Visual Basic Script) calls the synchronous translator 30. Software developers skilled in the art will recognize that many other programs can be written in accordance with the IDL to take advantage of the synchronous translator 30.

In contrast, when a client 11 sends a translation request that requires use of the asynchronous translator 32, the translation engine 16 will perform a translation in the background, so that the client 11 can perform other operations while the translation engine 16 is working. While the client 11 is performing other operations the translation engine 16 keeps the client 11 informed of the progress of the translation by providing information regarding this progress. The translation engine 16 can send this information directly to the client 11, or the client 11 can poll the translation engine 16 for this information.

When the client 11 sends a translation request to the asynchronous translator 32, the client 11 also provides the asynchronous translator 32 with a text to translate and a place to write the results. The client 11 provides two additional interfaces for this purpose: the translation source interface 96 and the translation sink interface 100. Each of these interfaces is specified by an IDL. The translation source interface 96 provides the text; the translation sink interface 100 consumes the translation (results are written to it). In other embodiments, the translation source interface 96 also can allow specification of annotations such as use-specified part of speech settings, HTML markup, etc. In still other embodiments, the translation sink interface 100 not only retrieves translated text, but also supplies part-of-speech information and dictionary source information from the translation engine 16.

In addition, when using the asynchronous translator 32, a client 11 also may want to know how to obtain progress information and know when the translation is done. In one embodiment, the client can provide a translation progress object 102 that is associated with the asynchronous translator 32 and receives notifications of progress. The translation progress object 102 has on interface defined by an IDL. In another embodiment, a client 11 can instead poll the asynchronous translator 32 for progress information.

Other features of the embodiment of FIG. 2 include the following. A sentence end finder 80, which has an interface defined by an IDL, attempts to find sentence boundaries in the text to be translated without performing a translation. An error handler 82 provides a notification mechanism to inform the client 11 if an error has occurred during translation, during the receipt of a translation request, during the receipt of preferences information, etc. Preferably, the error handler 82 is implemented using a call back interface defined by an IDL for specifying errors, providing severity information, and providing numerical error identifications. In other embodiments, an error call back object is associated with another object that operates in accordance with a distributed object protocol, such as the asynchronous translator 32, a client 11, or a user interface 24.

In some embodiments, the translation engine 16 provides a response to the client 11 in accordance with information relating to a desired format of the translation and in some of these embodiments, the client 11 is notified if a desired format is unavailable. In some embodiments, the translation system 10 further comprises an alternate word server 34 for locating at least one alternate word choice for a given translation request and communicating that alternate word choice to the client 11 and translation engine 16. In some embodiments, the system translates Japanese to English or vice-versa. In some embodiments, the translation engine can include a translation server for receiving information related to the desired format of the translation. Many other types of translation system features, such as custom dictionaries, spell checkers, background (asynchronous) translators, immediate (synchronous) translators, error checkers, etc., can also be included, as will be understood by those of ordinary skill.

Referring to FIG. 3, in another embodiment the translation engine further comprises storage 18 that may include one or more areas of disk (e.g., hard, floppy, optical, etc.) and/or memory (e.g., RAM) storage, or the like. The storage 18, in some embodiments, is used to store input textual information in a source natural language, output textual information in a target natural language, and all sorts of information used or useful in performing the translation including one or more dictionaries, domain keywords, grammar rules, and other databases.

FIGS. 4 and 5 illustrate two embodiments of the present invention that take advantage of the interoperability offered by the distributed object protocol employed in the present invention. It should be understood that in both FIG. 4 and FIG. 5, that all of the clients 11 and translation engines 16 operate in accordance with a distributed object protocol.

FIG. 4 illustrates an embodiment of the invention having a client 11 and a plurality of translation engines 16. Each translation engine 16 can receive a different translation request using a distributed object protocol from the client and produce a response corresponding to a translation of the text of the translation request. The response is sent using a distributed object protocol to the client.

Thus, for example, in FIG 4, a user of client 11 might have a given piece of text that is in English and needs to be translated into Japanese. The user also wants to be able to tell how accurate the translation is. The user inputs the text and client 11 formulates a translation request that is sent to one of the plurality of translation engines 16. In this case, the translation request is sent to a translation engine 16 that performs English-Japanese ("E-J"). Because this system can access multiple translation engines 16 simultaneously, the system can forward the response generated by the E-J translation engine as another translation request that is sent to a Japanese to English ("J-E") translation engine. Then, both the E-J and J-E translation engines provide responses back to client 11. The user at the client 11 can compare the response provided by the J-E translation engine (which translated the resultant translated text [Japanese] back to English) to the original English text, to verify the accuracy of the E-J translator. Such a technique is beneficial in helping a user to detect minor translation errors that occur when translation engines do not recognize the idioms in a particular language. Because these errors can result in a translation having an unintended meaning, it is very helpful to be able to access multiple translations simultaneously, regardless of the peculiarities of the particular translation engine.

The ability of this embodiment of the invention to remotely host a multitude translation engines, regardless of their origin, has other advantages as well. For example, in one embodiment the clients 11 may test a remote translation engine 16 using the user interfaces of their choice without the expensive download of the entire translation engine (often tens of megabytes or more).

In another example, in FIG. 4, a user of client 11 might have a given piece of text that is in English. The user wants to translate this text into a plurality of different languages, for example Japanese, Spanish, and German. The client 11 need only formulate one translation request. The client can then send this same translation request to any one (or more) of the translation engines 16 without having to customize the translation request for each translation engine 16. The translation engine 16 receiving the translation request sends the client 11 a response corresponding to a translation of the text. Thus, the client 11 could receive Japanese, Spanish, and German translations of a given piece of English text, from different translation engines 16, using just one translation request.

Embodiments of this aspect of the invention can include the client further comprising a user interface having a translation service for accepting translation requests and the distributed object protocol that the client and plurality of translation engines communicate with supporting a plurality of translation engines. In another embodiment, the distributed object protocol used by the plurality of translation engines and the client to communicate operates in accordance with the COM standard. In another embodiment, the distributed object protocol used by the plurality of translation engines and the client to communicate operates in accordance with the DCOM standard.

FIG. 5 illustrates an embodiment of the invention having a plurality of clients 11 and a translation engine 16. In this embodiment, the translation engine 16 is remotely located from one or more of the plurality of clients 11. Any one or more of the clients 11 can send a translation request to the translation engine 16 and receive a response from the translation engine corresponding to a translation of the text of the translation message.

This embodiment is useful especially when the client 11 is running on a user computer that does not have sufficient memory or processing power to have a translation engine 16 on it. This embodiment also is useful for multiple "on the fly" translations requested by a multiple of remote clients 11. For example, several employees of a company each may be on business trips, but are carrying with them systems such as palmtop computers, personal digital assistants (PDA), and the like. Users of such systems have the ability to communicate with remote computers via satellite and cellular connections. In accordance with this embodiment of the invention, each user's portable system can be a client 11 communicating with the same remotely located translation engine 16, sending the translation engine 16 multiple translation requests. Because each of the clients 11 communicates using a distributed object protocol, and because the translation engine 16 sends a response using a distributed object protocol, every client 11 will be able to get translations from the translation engine regardless of the type of client 11 (e.g. PDA, laptop, palmtop) or application (database, spreadsheet, etc.) that the client 11 is running.

One skilled in the art will recognize that all of the embodiments applicable to the aspects of the invention involving a single client and a single translation engine are likewise applicable to embodiments of the invention involving multiple clients and/or multiple translation engines. Many other types of translation system features, such as custom dictionaries, translation options, spell checkers, etc., can also be included, as will be understood by those of ordinary skill.

FIG. 6 shows a flow chart illustrating the steps taken in a disclosed embodiment of a method of the present invention. The steps will be illustrated using the example of an English to Japanese translator. It should be understood that the present invention is not to be limited to any particular languages. Referring to step 48, the client receives input text in a first language. This text could come from a word processing document, voice input, a string of characters to be input by a user operating a keyboard, etc. For example, the following sentence could be input to the client:

"Do you speak English?"

Next, in steps 50 and 52, the client generates a translation request from the input text and sends it to a translation engine in accordance with a distributed object protocol. In some embodiments, the translation method includes the steps of providing to the translation engine information relating to a desired format of the translation, and in some of these embodiments, the translation method includes the steps of notifying the client if the translation engine cannot perform a translation in accordance with the information. Thus, step 52 optionally could further include the steps illustrated in FIG. 7. In this example, the text string above would first be scanned for irregularities (such as typographical errors), as illustrated in step 64. The source text also could include annotations, HTML markups, sentence boundary mark-ups, Part -of- Speech settings, etc., for which the translation engine could scan. Next, referring to step 66, the user can select the desired translation format. In this example, the translation format desired is to select a Japanese translation that results in English alphabetic characters (romaji) instead of kanji characters. Next, referring to steps 68 and 70, the translation preferences object is sent to the translation engine to tell the translation engine the configured attributes of the translation. In one embodiment the translation preferences object is sent as part of the translation request. In another embodiment the translation preferences object may be sent independently of the translation request.

Referring again to FIG. 6, in step 54 the translation engine receives the translation request using a distributed object protocol. Referring to step 56, the translation engine then translates the text of the translation request to the second language in accordance with any selected and available translation options and in accordance with any selected translation preferences. In some embodiments the method can further comprise the step of notifying the client if an error has occurred during any of the steps. One skilled in the art will recognize that additional standard translation steps, such as translating in accordance with a particular dictionary, translating while the client is performing other tasks, translating in accordance with a particular format, etc., can be used in accordance with this aspect of the invention. Thus, for example, step 56 optionally can include the steps shown in FIG. 8. Referring to step 76 of FIG. 8, the client can be notified if an error has occurred during translation. This can occur, for example, if the translation engine encounters a character for which it has no translation. In the previous example message, if the translation engine did not have an appropriate translation for the question mark at the end of the text string, it would notify the client of this error. In one embodiment notification occurs immediately. In another embodiment, the notification forms part of the response sent to the client.

Another feature of the notification mechanism of the present invention is that the notification mechanism in accordance with the invention has an interface defined by an IDL. Thus, reporting of error messages is standardized between the client and the translation engine.

For example, in one embodiment there is a callback interface, as is understood by these skilled in the art, for specifying errors, providing severity information, and providing numerical identifiers for the errors. In one embodiment error callback objects are associated with objects (such as objects associated with asynchronous translation) that generate errors. In another embodiment the error callback objects can be provided as an argument to other steps. In one embodiment the client user interface implements this interface with a dialog box. However, it should be understood that any user communication that is capable of stating the error may be used to notify the user of errors.

In another embodiment implementing this notification mechanism, numerical error identifiers are associated with custom user interface language strings, to provide error messages into the user interface application.

In step 78 of FIG. 8, the translation engine in one embodiment can notify the client of the progress of the translation. This is helpful if the text to be translated is quite long and/or if the client has selected a translation option such as asynchronous translation (which performs a translation in the background while freeing the client to perform other tasks).

Referring again to FIG. 6, after the translation step 56 is completed, the translation engine generates a response (step 58) and sends the response to the client (step 60) using the distributed object protocol. Using the previous example, for the translation request:

"Do you speak English?"

the client receives a response comprising text in the second language. For an English to Japanese translation engine, in accordance with the option of using an English alphabetic characters, the text received would be:

"Eigo o hanashi mas ka?"

Any of the embodiments of the modular language translation system described herein, including all of the functionality described herein, can be provided as computer software on a computer readable medium such as a diskette or an optical compact disc (CD) for execution on a general purpose computer (e.g., an Apple Macintosh, an IBM PC or compatible, a Sun Workstation, etc.).

Variations, modifications, and other implementations of what is described herein will occur to those of ordinary skill in the art without departing from the spirit and the scope of the invention as claimed. Accordingly, the invention is to be defined not by the preceding illustrative description but instead by the spirit and scope of the following claims.

What is claimed is: